# PML Assignment 1

***Li Ying***

***16 June 2015***

# Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit it is now possible to collect a large amount of data about personal activity relatively inexpensively. These type of devices are part of the quantified self movement – a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, ** but they rarely quantify how well they do it **.

In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

# What you should submit

The goal of your project is to predict the manner in which they did the exercise. This is the **classe** variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

1. Your submission should consist of a link to a Github repo with your R markdown and compiled HTML file describing your analysis. Please constrain the text of the writeup to < 2000 words and the number of figures to be less than 5. It will make it easier for the graders if you submit a repo with a gh-pages branch so the HTML page can be viewed online (and you always want to make it easy on graders :-).
2. You should also apply your machine learning algorithm to the 20 test cases available in the test data above. Please submit your predictions in appropriate format to the programming assignment for automated grading. See the programming assignment for additional details.

# Reproducibility

Due to security concerns with the exchange of R code, your code will not be run during the evaluation by your classmates. Please be sure that if they download the repo, they will be able to view the compiled HTML version of your analysis.

# Download Data

```
setwd("~/Documents/## Github Repo/Human-Activity-Recognition_Machine-Learning")
library(caret)
url1<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url2<-"https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# Create a folder called data
if(!file.exists("data")){dir.create("data")}

# Download training and testing file to data
#download.file(url1,destfile="./data/training.csv",method="curl")
#download.file(url2,destfile="./data/testing.csv",method="curl")

# Read data into R
training<-read.csv("./data/training.csv")
testing<-read.csv("./data/testing.csv")



set.seed(3456)
inTrain <- createDataPartition(training$classe, p=0.7, list=FALSE)
training <- training[inTrain,]
testing <- training[-inTrain,]
```

# Explore data

```
length(names(training))
```

```
## [1] 160
```

```
table(sapply(colnames(training),FUN=function(x){class(training[,x])}))
```

```
##
##  factor integer numeric
##      37      35      88
```

```
# table(apply(training,2,class))
```

Missing values

```
# Compute the sum of NA of each column
rowNA<-apply(is.na(training),1,sum)
table(rowNA)
```

```
## rowNA
##     0    67
##   280 13457
```

```
# Therefore, we have 406 out of 19622 observations that is complete in every column

colNA<-apply(is.na(training),2,sum)
table(colNA)
```

```
## colNA
##     0 13457
##    93    67
```

```
# There are 93 columns that have no NAs
```

# Reduce dataset

Take away the NA columns

```
numColNA<-data.frame(col_name=names(training),NAcount=colSums(is.na(training)),row.names=NULL)
validColNames<-subset(numColNA,NAcount==0,row.names=NULL)$col_name
noNA<-training[,names(training)%in%validColNames]
```

Remove varaibles with near zero variance

```
nzv<-nearZeroVar(noNA)
noNA_zv<-noNA[,-nzv]
```

Deal with variables of high correlations

```
table(sapply(colnames(noNA_zv),FUN=function(x){class(training[,x])}))
```

```
##
##  factor integer numeric
##       3      29      27
```

```
# cor() does not work on factors

# get the column names that is not character
x<-factor()
for (i in names(noNA_zv)){x<-c(x,class(noNA_zv[,i]))}
colClass<-data.frame(col_names=names(noNA_zv),class=x)

noFactorNames<-subset(colClass,!class=="factor")$col_names

# convert rest data to interger
noFactor<-noNA_zv[, noFactorNames]
for (i in 1:dim(noFactor)[2]){
  noFactor[,i]<-as.numeric(noFactor[,i])
}

cor_training <- cor(noFactor)
# find highly correlated descriptors, return a vector of intergers correspondng to columns
to remove to reduce pair-wise correlations
corrDescr <- findCorrelation(cor_training, cutoff = 0.75)
noFactorNames[corrDescr]
```

```
##  [1] gyros_arm_y          accel_dumbbell_x     accel_belt_y
##  [4] magnet_arm_y         accel_forearm_x      pitch_dumbbell
##  [7] gyros_arm_z          pitch_forearm        raw_timestamp_part_1
## [10] pitch_belt           raw_timestamp_part_2 magnet_arm_x
## [13] pitch_arm            magnet_arm_z         magnet_belt_y
## [16] gyros_belt_x         accel_arm_x          gyros_forearm_y
## [19] total_accel_belt     total_accel_dumbbell magnet_forearm_y
## 59 Levels: accel_arm_x accel_arm_y accel_arm_z ... yaw_forearm
```

```
noNA_zv_cor<- noNA_zv[,!names(noNA_zv) %in% noFactorNames[corrDescr]]
```

We can see that there are a factor variable called user_name, a variable X which is simply the row number and finally num_window. Let's remove them since they are irrelevant.

```
noNA_zv_cor <- noNA_zv_cor[, !(names(noNA_zv_cor) %in% c('user_name', 'X', 'num_window'))]

table(sapply(noNA_zv_cor,class))
```

```
##
##  factor integer numeric
##       2      15      18
```

# Training

Because `class` is a factor variable we will need to train a classifier and since it has more than two levels, we cannot use Generalized Linear Models (glm).

Instead we have to use more advanced classifiers. More specifically classification and regression trees (CART), stochastic gradient boosting and random forest. All classifiers will be trained using 5-fold Cross Validation to reduce model overfitting and to estimate the out of sample error during training. We expect this estimate to be close to the final out of sample error that we will calculate later on using the test dataset. Our metric will be accuracy.

```
trControl <- trainControl(method="cv", number=5)
```

# 1. CART (rpart)

```
noNA_zv_cor_rpart <- train(classe ~ ., data = noNA_zv_cor, method='rpart', trControl = trC
ontrol)

noNA_zv_cor_rpart$results[1,]
```

```
##           cp  Accuracy     Kappa AccuracySD    KappaSD
## 1 0.03756824 0.4873625 0.3275534 0.05360248 0.07217154
```

### Result

```
pred_rpart <- predict(noNA_zv_cor_rpart, testing)
cm_rpart <- confusionMatrix(pred_rpart, testing$classe)
cm_rpart
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1085  520  345  418  160
##          B    0    0    0    0    0
##          C  104  256  377  248  247
##          D    0    0    0    0    0
##          E    4    0    0    0  370
##
## Overall Statistics
##
##                Accuracy : 0.4432
##                  95% CI : (0.4279, 0.4585)
##     No Information Rate : 0.2886
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2619
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9095   0.0000  0.52216   0.0000  0.47619
## Specificity            0.5094   1.0000  0.74941   1.0000  0.99881
## Pos Pred Value         0.4292      NaN  0.30601      NaN  0.98930
## Neg Pred Value         0.9328   0.8123  0.88112   0.8389  0.89176
## Prevalence             0.2886   0.1877  0.17465   0.1611  0.18795
## Detection Rate         0.2625   0.0000  0.09119   0.0000  0.08950
## Detection Prevalence   0.6115   0.0000  0.29802   0.0000  0.09047
## Balanced Accuracy      0.7094   0.5000  0.63579   0.5000  0.73750
```

Out of sample error measured by accuracy is 0.4233. During training, this was estimated to be 0.5293. This classifier does not perform well.

# 2. Random Forest (parRF)

```
library(randomForest)
noNA_zv_cor_rf <- randomForest(classe ~ ., data = noNA_zv_cor)

noNA_zv_cor_rf
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = noNA_zv_cor)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 0.53%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3898    3    5    0    0 0.002048131
## B   13 2639    6    0    0 0.007148232
## C    0   13 2374    9    0 0.009181970
## D    0    0   13 2237    2 0.006660746
## E    0    0    2    7 2516 0.003564356
```

# Result

```
prediction3<-predict(noNA_zv_cor_rf ,testing)
confusionMatrix(prediction3,testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1193    0    0    0    0
##          B    0  776    0    0    0
##          C    0    0  722    0    0
##          D    0    0    0  666    0
##          E    0    0    0    0  777
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9991, 1)
##     No Information Rate : 0.2886
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000    1.000
## Specificity            1.0000   1.0000   1.0000   1.0000    1.000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Prevalence             0.2886   0.1877   0.1746   0.1611    0.188
## Detection Rate         0.2886   0.1877   0.1746   0.1611    0.188
## Detection Prevalence   0.2886   0.1877   0.1746   0.1611    0.188
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000    1.000
```

The accuracy is close to 100%. In the training set, out of sample rate is 0.58%

# Summary

To summarize we used the caret R package to split the data into two datasets for training and testing. We then preprocessed the training dataset using several techniques (missing values, near zero values, linear correlations) and managed to reduce the number of variables to 35. Then we trained three different classifiers using the CART, Random Forest and Boosting methods (unsucessful attempt). From these three classifiers, the one based on random forests performed significantly better with accuracy close to 100% on the test dataset.